Workshop Proceedings of the 11th International Conference on Intelligent Environments
D. Preuveneers (Ed.)

doi:10.3233/978-1-61499-530-2-46

An Approach Addressing Service Availability in Mobile Environments

Gabriel GUERRERO-CONTRERAS [a], Sara BALDERAS-DÍAZ [a]
Carlos RODRÍGUEZ-DOMÍNGUEZ [a], Aurora VALENZUELA [b] and
José Luis GARRIDO [a]

[a] *Software Engineering Department, E.T.S.I.I.T, University of Granada*
C/ Periodista Daniel Saucedo Aranda S/N, Granada, Spain
gjguerrero@ugr.es, sarabd@correo.ugr.es, carlosrodriguez@ugr.es, jgarrido@ugr.es
[b] *Dpt. of Forensic Medicine, Toxicology and Physical Anthropology,*
Faculty of Medicine, University of Granada
Avenida de Madrid 11, 18012 Granada, Spain
agarach@ugr.es

Abstract. Mobile Cloud Computing paradigm has arisen as a major proposal to address collaboration support in working environments. Particularly, this paradigm has proven to be useful in emergency scenarios, education or tourism. However, these environments are commonly based on dynamic network topologies, which imply unstable connections (disconnections and network partitions). In consequence, the availability of the services can be compromised. Therefore, approaches based on the Service Oriented Architecture (SOA) are insufficient and they must be complemented with techniques and methods of Autonomic Computing, in order to ensure the quality attributes of the system against context changes. In this work, a self-adaptive architecture is proposed in order to address the availability of the services deployed in dynamic network environments. This architecture has been designed to provide a common basis for collaborative mobile systems. The architecture follows a component-based design, and it provides a distributed approach to support the dynamic replication and deployment of services. Further, an example scenario based on a real Mobile Forensic Workspace is described to show the applicability of the proposal.

Keywords. Context-awareness, autonomic computing, dynamic network topologies, mobile environments, Service Oriented Architecture (SOA)

1. Introduction

The Service Oriented Architecture (SOA) proposes a modular distribution of the functionalities of a system through services [1]. It provides the foundations to build an interoperable and scalable system thanks to the use of standard protocols and service composition. However, SOA itself is not sufficient to be able to operate in dynamic network environments [2], such as *Mobile Cloud Computing* [3]. Such environments pose new features, and, if they are not correctly addressed, they may have a significant impact on the quality attributes of the services [4]:

- *Energy constrained.* The energy supply is one of the main weaknesses of the mobile devices. Some approaches to guarantee certain quality attributes in this kind of system are often in conflict with the energy-aware performance. For instance, the availability of a service is directly proportional to its number of replicas. However, an intensive replication will require intensive energy consumption. Therefore, it is necessary to provide an energy-aware architecture that balances the performance of the system with efficient energy consumption.
- *Dynamic network topologies.* The topology of a mobile network changes frequently and unpredictably, owing to the mobility of the nodes that makes up the network. Further, the nodes may be switched off or may be disconnected (temporarily or permanently). Since these networks are typically multi-hop, this usually leads to link failures, route changes or even network partitions.
- *Dynamic demand patterns.* The set of clients of a service is increased/decreased over time. Additionally, the location of these clients may change. These circumstances can result in bottlenecks and an inefficient use of the available bandwidth.
- *Heterogeneity.* A wide range of computer-based subsystems (laptops, smartphones, wearable devices, sensor networks, etc.) are being used and integrated for building current advanced systems. The performance, average reliability, available technologies (such as GPS, NFC or Bluetooth), and capabilities of each node may vary.

These features pose a challenge for engineers and developers of software architectures, who must make suitable architectural design decisions to address them correctly. For instance, the reliability and performance of distributed applications are critically conditioned by the placement of the services in the distributed system [5]. Nevertheless, the conditions of energy, topology, demand patterns, and heterogeneity vary over time, i.e., they are dynamic features. Therefore, in order to address the availability of services, some decisions and associated techniques, such as deployment, replication, and migration of services should be applied in run-time.

As a result, self-adaptive architectures have been gaining importance in the research community [6]. A self-adaptive architecture has been complemented with self-* features (i.e., self-healing, self-configuration, self-optimization and self-protection [7]), and it has the capability of reducing the impact of context changes in the quality attributes of the system. This kind of architectures has a substantial potential in different application domains. For instance, in the forensic domain can be found different scenarios: natural disasters, accidents, terrorist attacks, etc., where the common network infrastructures are not available and the forensic experts need apply action protocols to support victim identification. In this context, the *Mobile Forensic Workspace* [8] intends to provide support to forensic experts in data collection and sharing using a mobile system. However, the features previously exposed may negatively affect the availability of the services, and thus hampering to achieve the ultimate goal of forensic experts.

This paper introduces a self-adaptive architecture, which aims to guarantee the availability of services in dynamic environments. This architecture has been designed to provide a common basis for mobile collaborative systems. It is based on previous findings [9], which have been validated through a preliminary evaluation on the Network Simulator 3[1]. The results obtained have allowed ratify some of the concepts previously pro-

[1] https://www.nsnam.org/.

posed and introduce new findings in the proposed software architecture. These findings are introduced at three different levels: (1) in the design of the architecture, the components of each subsystem are identified and defined; (2) in the communication between the components, the events that trigger the adaptation are described; and (3) in the management of the network topology, a new method to identify the position of the nodes within the network are proposed. The architecture follows a component-based design, and provides three main subsystems: the Monitor Subsystem, which is responsible for monitoring events in the context of the device that can affect to the availability of a service; the Context Manager Subsystem, which store information about the own device and other nodes of the network; and the Replica Manager Subsystem, which carries out the dynamic replication and deployment of services through a fully distributed approach.

The rest of this paper is organized as follows. Section 2 presents a study of the main issues that have been addressed in the literature in order to improve the availability of services in dynamic environments. Section 3 introduces a self-adaptive software architecture to support the dynamic replication and deployment of services. In Section 4, an example based on the Mobile Forensic Workspace is described to show how the approach helps to improve the availability of the services deployed in that system. Section 5 provides a brief discussion about the proposal. Finally, Section 6 draws some conclusions and outlines the plans for future research.

2. Related Work

In 2003, IBM proposed a reference model for autonomic control loops, called MAPE-K loop (Monitor, Analyze, Plan, Execute, Knowledge) [10]. This model has been widely used as base to define autonomic systems, including systems that require dynamic replication and deployment of services. Regarding this kind of systems, two steps in the adaptation loop have a major impact on their performance: when replicate and where place the replica.

There exist different events that can trigger the creation, migration or deletion of replicas, for example, the battery of the host device is running out, or the device switches off, the demand for the service increases, etc. Between these events, the prediction of a network partition can be highlighted. A network partition can affect to the availability of a service and consistency of the shared information [11]. The fact of predicting a partition allows taking the necessary actions to ensure the consistency of the shared information and the availability of the service, while the connection is still available. Chandrakala et al. [12] propose a predictive algorithm based on the position of the nodes, their speed travel and range. When a partition is predicted, the node where the replica will be placed is chosen by taking into account the distance from the source node (i.e., the node that have a copy of the service and from where the replica will be created and sent), its battery and its storage capacity. In [13], TORA routing protocol is used in combination with an estimation of the residual link lifetime of wireless links. When a device predicts a partition, this device will host a replica of the service, regardless of its characteristics.

In collaborative systems, the partition prediction algorithms may be relegated to second place. In this kind of systems [14,15], the set of services is well-known in deployment time, thus an approach based on hibernation [16,17] in which the replicas of the services can be deployed in each device is feasible. On the one hand, the advantages of

this approach are as follows: better response times; the service does not have to be sent in real time, hence, there is a more efficient bandwidth and energy use; and also the system is less dependent on the available technology, because the requirements on bandwidth are less stringent. On the other hand, the system will be less flexible, since new services cannot be introduced in the network at run-time.

Moreover, in large scale systems [2] the scalability is a main objective for software architects. Device clustering methods are applied in order to turn a distributed network into a set of interconnected local clusters that can be dealt with individually like a centralized network. In this way the management of the network is simplified, achieving scalability under a "divide and rule" approach. Dustdar et al. [16] create device clusters on the basis of the distance between mobile devices. However, in [18,19] is shown that the travel speed of devices is a better measure to create clusters of mobile devices. In [19], service replicas will be created when too many requests are made to a service from an external group. Hence, this proposal intends to achieve the property of localized scalability [20], reducing the communications between distant entities, in order to safeguard the bandwidth and the energy of the system. In [16,19], the device that will host the new replica is chosen by considering its computational capabilities (battery, CPU, memory, etc.), without taking into consideration either its current workload or the network topology. Consequently, the resources of the host device can be quickly depleted.

Finally, Hamdy et al. [17] propose a replication protocol based on the interest of devices in the use of the service. When an application needs to access a service and there is not a replica of that service in its neighbourhood, a replica of the service will be created and deployed in the same device in which the application is hosted.

Generally, these are ad-hoc solutions developed for specific application domains, and they are based on an implicit, and often restricted, context model. The definition and use of an explicit context model can facilitate the adaptability and reusability of the proposal. The possibility of extending the model according to the particular requirements of a scenario would provide a more effective solution in terms of performance to such scenario.

3. A Self-Adaptive Software Architecture

The proposed software architecture has two main objectives: (1) provide a reusable and adaptable base for collaborative support systems, (2) enhance the availability of services in dynamic environments through an adaptive replication and deployment approach.

The architecture has been initially designed to support medium size work groups, which allows a hibernation approach. Thus, it follows a logic approach to the replication and deployment of services. Further, in order to provide a fully distributed approach, the different nodes of the network must coordinate themselves at run-time to decide which of them will be the active replica.

In order to provide an adaptable and reusable software architecture a component-based design has been followed [21]. The architecture can be divided into five main subsystems (Figure 1): the *Monitor Subsystem*, which is responsible for monitoring events in the context that can affect the availability of a service; the *Context Manager Subsystem*, which processes and stores that information; such information will be used by the *Replica Manager Subsystem*, which encapsulates the adaptation logic regarding the repli-

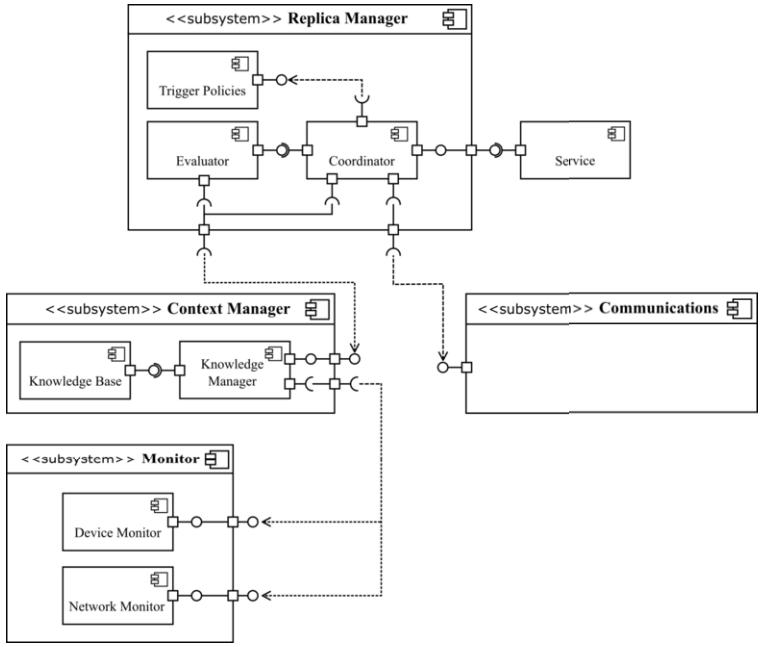

Figure 1. Subsystems for the self-adaptive software architecture approach.

cation and deployment of the service replicas; the *Communications Subsystem*, which allows the *Replica Manager Subsystem* to communicate with other nodes of the network; and finally the *Service* itself, which will be a passive or active replica according to the decision of the *Replica Manager Subsystem*. These subsystems are described in depth in the following sections.

3.1. Monitor Subsystem

The *Monitor Subsystem* encompasses a set of monitoring components, which are sensing the context of the device in order to detect potential events that could affect the availability of a service. To monitor the context is a costly operation in terms of energy and bandwidth. Nevertheless, the performance and response time of the adaptation system depend on the precision of this action. Therefore, to provide an appropriate balance between the cost and the precision, through the sampling frequency, is required.

In the proposed architecture, device and network capabilities are monitored. The information about the device capabilities is local information that usually can be obtained easily. However, a dynamic network topology requires a continuous monitoring because of constant changes that occur. Further, the cost increases exponentially with each hop in the network. Accordingly, the scope of the monitoring will be limited.

There are different approaches to monitoring a network topology. The majority of those approaches are based on the position of the nodes and their travel speed. This requires using the GPS system and exchange such information constantly, which have a heavy impact on the energy of the device. As a more efficient alternative, in this work is proposed to use the information provided by the routing protocol, such as *Optimized Link State Routing Protocol* (OLSR) [22], to estimate the network topology.

The routing protocol builds and provides the routing tables with information about the reachable nodes, and for each node the gateway and the number of hops. Through the direct connections of a node, its position within the network topology can be approximated. For instance, a node with four direct connections will be in a more centric position within the node group, and therefore will be a more stable node than other ones with only one direct connection. This approach is more efficient than GPS-based ones, but less precise. Still, the component-based design allows exchange the *Network Monitor* component, adapting the system at run-time according to current conditions [23].

3.2. Context Manager Subsystem

The *Context Manager Subsystem* is responsible for processing and storing the information received from the monitors of the device. This information will be used by the *Replica Manager Subsystem* in order to adjust the deployment of the services according to the changes produced in the execution context. This information will be provided in two different ways: (1) under a Publish-Subscribe paradigm, where the *Context Manager* notifies, through a push-based communication model, the *Replica Manager* about events of its interest. For instance, an event will report the battery level every time that the battery decreases a 5%, or an event will be published when the active replica is not reachable; and (2) under a Request-Response paradigm, when, for example, the *Replica Manager* requests the *Context Manager* about information to evaluate the adequacy of the node. The combination of both paradigms is usually known as SOA 2.0 [24,25] (a.k.a. advanced SOA), in which services are not just passive entities, but also they are able to receive and generate events proactively.

Although, all the nodes share the same data model, the information stored by the *Context Manager Subsystem* is mainly local to device (battery consumption, number of direct connections to other devices, remaining storage capacity, etc.) and only the evaluation of the node is shared with the rest of the system, in order to reduce the bandwidth consumption. Hence, each node maintains a list with the evaluation of each reachable node in the network.

The importance of the different context aspects that may influence the dynamic replication and deployment of services will depend on the particular application domain. However, the following context aspects have been identified as relevant to provide a solution, regardless application domain (Figure 2): device features (e.g., battery, memory, processing capacity, etc.), network topology (e.g., client-service distance) and service requirements, i.e., how the device features match the requirements of each specific service.

- *Device features.* A device feature can be a simple feature, such memory or storage, or a compound feature, such remaining battery. That is, a compound feature is derived from two or more simple features. For example, remaining battery could be calculated from screen brightness, CPU usage, phone signal strength, and other features. It is important to take into consideration the device features in order to choose the devices with best performance to host services. Also, the current usage of these resources must be taken into consideration to deploy a service replica (remaining battery, CPU or memory, among others).
- *Service requirements.* Although there is cross-cutting information that affects in a similar way the deployment of all services of the system, such as the device battery or client-server distance, there is other information that affects in a greater or

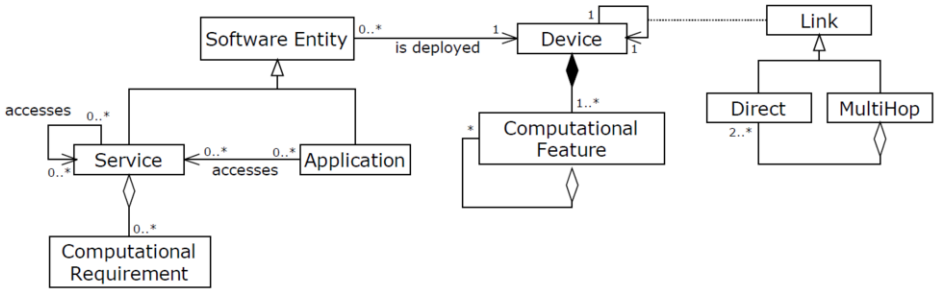

Figure 2. A context model for self-adaptive service replication and deployment in dynamic environments.

lesser measure in relation to the service requirements. For example, one service could need high processing power while other could need high storage capacities. For this reason the services must specify theirs *Computational Requirements* independently [14]. The service developer must indicate what features must be provided by the devices in which the service could be deployed, through a set of condition statements (e.g., "free storage space equal or greater than 1GB"). This would allow the system to distribute the workload among the devices of the network, providing the appropriate deployment of the services. Moreover the number of clients (i.e., applications and other services), dependencies between services (i.e., regarding service composition) and number of replicas of the service, are also interesting context information to provide an efficient service deployment solution.

- *Network topology.* Each device has two kinds of *Links*: *Direct links* (one-hop) and *Multihop links*. Each *Multihop link* is made up of two or more direct links between devices. In order to simplify the management of the network topology, the model only considers the best path between two devices, but it is independent of its calculation (fewest hops, higher bandwidth, best stability, etc.).

3.3. Replica Manager Subsystem

The *Replica Manager Subsystem* encapsulates the adaptation logic regarding the replication and deployment of the service replicas. In order to provide a fully distributed solution, each service replica has a *Replica Manager Subsystem*. It consists of three main components:

- *Evaluator.* This component implements the evaluation function that will be used by the Coordinator component to evaluate the suitability of the node to host a specific service. This evaluation is based on the information provided by the Context Manager, and the weight of each context feature will depend on the application domain and the specific situation. For instance, if the battery of a device is running low, the evaluation function can dynamically change to assign more priority to this context feature. Therefore, a developer could define different Evaluator components to address different situations occurring in a specific application domain.

- *Trigger Policies.* Through this component, the developer can adapt the system to the particular requirements of a specific application domain. In it, through a rule-based system are described the trigger policies, which are related with the reception of events through the Context Manager. This information will help the Coordinator component to know when adapt the deployment of the service could be needed. For example, by way of these rules can be indicated: *replicate* a service when the resources of the current host device are running out (e.g., the battery power is below 20%), the topology of the network changes (e.g., a partition is predicted, or the active replica is not already reachable), or new clients appear; *migrate* a replica when the distribution of the clients changes, or a better device to host the replica is discovered; or *hibernate* a replica when the number of clients is reduced. As with the Evaluator component, different trigger policies could be necessary to address different situations in the same system.
- *Coordinator.* When, through trigger policies, it is detected a change in the context that could affect the quality attributes of the service, the Coordinator component is responsible for coming to an agreement with the rest of the replicas deployed in the system. The objective of this coordination is to know if a better deployment exists and, if so, to establish what will the active replica. As the proposed architecture follows a distributed approach, a consensus system is followed. Each Coordinator evaluates itself and sends this evaluation to the other reachable nodes. At the same time, it receives the evaluation from the other nodes. Using this information, each node gives its vote to the node that it considers most appropriate to host the replica, and the most voted node will host the replica. This procedure provides robustness against message loss, owing to it is not necessary to the proper functioning that all nodes have an identical list of evaluations, and the node election process only requires that it receives half plus one of the votes. The adaptation process can be initiated by any Coordinator as they have local information that is not accessible by the rest. Once that one Coordinator requires for a check of the current deployment of the service, the rest of the Coordinators follow the initiative and start the vote process.

4. Example

The *Mobile Forensic Workspace* [8], which allows forensic experts to exchange information in real-time for data sharing purposes in case of natural disasters, accidents, terrorist attacks, etc., is helpful to show the need and usefulness of the proposed architecture. The Figure 3 depicts a hypothetical scenario where three members of a forensic team are gathering preliminary evidences of two victims. In the scenario two kinds of devices can be found: (1) a laptop deployed statically in a police car; and (2) three mobile devices (one for each team member). Additionally, different services can be found (communication, image repository, victim's information repository, etc.). In the case of Image Repository service, each forensic expert can take several pictures of an evidence with his mobile device; different forensic experts can take photos for the same evidence; or the same forensic expert can take photos for evidences that belong to different victims. Moreover, the forensic experts can add annotations to these pictures, which can be related with other pictures. Therefore, the Image Repository service must keep an ordered and consistent

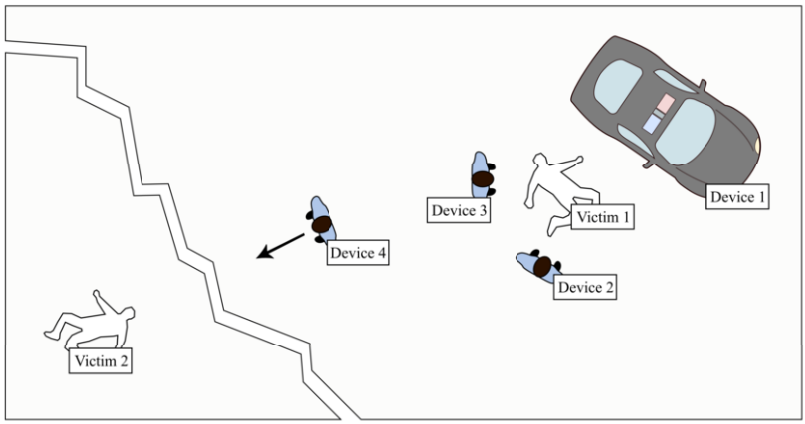

Figure 3. A hypothetical scenario within the Mobile Forensic Workspace case study.

set of this information, and at the same time it must provide high availability, to allow forensic experts to access and share pictures about the evidences found.

In this scenario, the proposed self-adaptive architecture can be useful to improve the availability of the Image Repository service. This is highlighted in the following situations:

1. Initially, all forensic experts are near the Victim 1. All the mobile devices are connected with the laptop located in the police car. Therefore, the active service replica (copy-primary protocol) is the one deployed in the laptop (*Device 1* in the Figure 3), as it presents better computational features.
2. The forensic expert of the Device 4 will move to the area where the Victim 2 is located. This area is out of the laptop coverage. The Monitor subsystem detects this situation, by way of a disconnection prediction method or because the active replica is not already reachable. Thus, it publishes the proper event that is received by the Replica Manager, through the Context Manager. The Replica Manager can manage this situation by means of its trigger policies and it starts the adaptation process. As the only replica available is itself, it will turn to active. At this point, the network is partitioned into two groups: (1) the Devices 1, 2 and 3, where the Device 1 hosts the active replica; and (2) the Device 4, that provides service to itself.
3. Later, the forensic experts of the Devices 2 and 3 move to the victim two area. The Context Manager of the Device 4 has been publishing events related with the battery power every time that it has decreased a 5%. At a given time, this event reports that the battery is under the 25%, therefore, the rule that reflect this situation in the trigger policies is activated. The Coordinator will search a better node in order to host a replica of the service. Following a distributed process, each replica will evaluate its adequacy to host the active replica, it will send this evaluation to their neighbours, and once that obtains the score list of the nodes, it will emit a vote for the best ranked node. The node with the half plus one votes will provide an active replica. Moreover, the active replica deployed in the laptop (Device 1 in the Figure 3) will have no clients, in this device there are no

applications that require the service. Therefore the Coordinator will hibernate the replica in order to save resources.

5. Discussion

The proposed architecture provides a base to support the dynamic deployment of services in dynamic network environments. This architecture helps to address the availability of the services for this kind of environments, and with it, the applicability of SOA-based approaches. Moreover, the management of the self-adaptation is carried out through a fully distributed approach. This provides robustness to the system against node disconnections or failures, since no node is indispensable. Besides, the consensus-based approach to elect the hosting node provides robustness against message loss.

Furthermore, together with the architecture, a context model is provided. It allows adapt the proposal to the specific requirements of an application domain. Additionally, the component-based design allows adapt dynamically the behaviour of the architecture. Generally, the proposals presented to improve the quality attributes of services in dynamic environments (see Section 2) are ad-hoc solutions developed for specific scenarios, and they are based on an implicit, and often restricted, context model. The presence of an explicit context model allows developers to customize or reuse the proposal.

The current proposal is designed to support medium or small size work groups. It is designed on the basis of a copy-primary scheme [26] in physical partitions. Nevertheless, in large scale networks this replication scheme could be inefficient. In large groups the active replica could easily become in a bottleneck, and therefore, a physical partition could need more than one active service replica. The creation of logical partitions provides scalability to system and it facilitates the management of a large scale network. This can be done through clustering techniques, and will be transparent for the adaptation system, as the adaptation logic to manage physical network partitions is transferable to manage logical partitions.

Finally, the proposed architecture follows a hibernation-based deployment approach. The set of the services is well-known at design time and their replicas are deployed in the devices before run-time. Such approach reduces the flexibility of the systems, as it is not possible to deploy a new service or introduce new devices at run-time. However, it can reduce the requirements about bandwidth and improve the response time of the adaptation process. Thus, the current architecture follows a logic approach to the replication and deployment of services. However, a hybrid approach could be provided, where well-known set of devices could be deployed in deploy time and also could be possible to add and deploy new services at run-time. This is a technical question, since the current implementation of the adaptation logic is transferable between both approaches, and the code mobility [27] can be implemented in a transparent way for the current adaptation process.

6. Conclusions and Future Work

In this work, a self-adaptive architecture has been presented. The architecture has two main objectives: (1) provide a reusable and adaptable base for collaborative support

systems; and (2) try to enhance the availability of services in dynamic environments through an adaptive replication and deployment approach. The management of the self-adaptation is carried out through a fully distributed approach, providing robustness to the system. This approach takes into account crosscutting context features, such as the network topology or the battery device. Moreover, the Mobile Forensic Workspace has been described. It illustrates the need and usefulness of the proposed self-adaptive software architecture.

At present time, a study of the proposed system in a network simulator is under development. This study will allow investigate the performance of the system regarding different configuration parameters (such as monitoring intervals, routing protocols, replication strategies, etc.), and to empirically compare the behaviour and the performance to other approaches. Preliminary results show a great increment in the availability of the services, with a uniform consumption of the battery of the devices of the network, and distributing uniformly the workload between the different nodes of the network. As future work, the self-adaptive architecture will be extended with clustering techniques, in order to address scalability in an integrated way.

Acknowledgment

This research is supported by the Centro Mixto Ministerio de Defensa (MADOC) and Universidad de Granada, and funded by Banco de Santander, under the granted project with Ref. PINS-2014-Cod-9; we thank TCol. Doctor Ignacio Álvarez de Cienfuegos (ISFAS Granada) for being part of the project team and for his active help in its progress.

This research is also related with the Project Ref. TIN2012-38600 granted by the Spanish Ministry of Economy and Competitiveness (FEDER), and the scholarship program FPU granted by the Spanish Ministry of Education, Culture and Sports.

References

[1] T. Erl, SOA: principles of service design, *Prentice Hall Upper Saddle River* (2008).

[2] P. Choudhury, A. Sarkar & N.C. Debnath, Deployment of Service Oriented architecture in MANET: A research roadmap, *9th IEEE International Conference on Industrial Informatics (INDIN)* (2011), 666–670.

[3] N. Fernando, S.W. Loke, & W. Rahayu, Mobile cloud computing: A survey, *Future Generation Computer Systems* **29** (2013), 84–106.

[4] C.B. Chandrakala, K.V. Prema & K.S. Hareesha, A Study of Location-Based Data Replication Techniques and Location Services for MANETs, *Advances in Computer Science, Engineering & Applications* (2012), 481–487.

[5] M.C. Little & D.L. McCue, The replica management system: a scheme for flexible and dynamic replication, *Proceedings of 2nd International Workshop on Configurable Distributed Systems* (1994), 46–57.

[6] D. Weyns & T. Ahmad, Claims and evidence for architecture-based self-adaptation: A systematic literature review, *Software Architecture* (2013), 249–265.

[7] J.O. Kephart & D.M. Chess, The vision of autonomic computing, *Computer* **36** (2003), 41–50.

[8] C. Rodríguez-Domínguez, K. Benghazi, J.L. Garrido, A.V. Garach, Designing a communication platform for ubiquitous systems: The case study of a mobile forensic workspace, *New Trends in Interaction, Virtual Reality and Modeling* (2013), 97–111.

[9] G. Guerrero-Contreras, C. Rodríguez-Domínguez, S. Balderas-Díaz & J.L. Garrido, Dynamic Replication and Deployment of Services in Mobile Environments, *New Contributions in Information Systems and Technologies* **353** (2015), 855-864.

[10] IBM Group and others, An architectural blueprint for autonomic computing, *IBM White paper* (2003).
[11] G. Guerrero-Contreras, J.L. Garrido, S. Balderas-Díaz & C. Rodríguez-Domínguez, Consistent Management of Context Information in Ubiquitous Systems, *7th International Conference on Internet and Distributed Computing Systems (IDCS)* (2014), 184–193.
[12] C.B. Chandrakala, K.V. Prema & K.S. Hareesha, Improved data availability and fault tolerance in MANET by replication, *IEEE 3rd International Advance Computing Conference (IACC)* (2013), 324–329.
[13] A. Derhab & N. Badache, A pull-based service replication protocol in mobile ad hoc networks, *European Transactions on Telecommunications* **18** (2007), 1–11.
[14] G. Guerrero-Contreras, J.L. Garrido, K. Benghazi Akhlaki, S. Balderas-Díaz & C. Rodríguez-Domínguez, Towards a Self-Adaptive Deployable Service Architecture for the Consistent Resource Management in Ubiquitous Environments, *Workshop Proceedings of the 10th International Conference on Intelligent Environments* (2014), 206–217.
[15] A. Neyem, S.F. Ochoa, J. Pino & R.D. Franco, A reusable structural design for mobile collaborative applications, *Journal of Systems and Software* **85** (2012), 511–524.
[16] H. Psaier, L. Juszczyk, F. Skopik, D. Schall & S. Dustdar, Runtime Behavior Monitoring and Self-Adaptation in Service-Oriented Systems, *Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)* (2010), 164–173.
[17] M. Hamdy, A. Derhab & B. König-Ries, A Comparison on MANETs Service Replication Schemes: Interest versus Topology Prediction, *Recent Trends in Wireless and Mobile Networks* (2010), 202–216.
[18] K.H. Wang & B. Li, Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks, *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)* **2** (2002), 1089–1098.
[19] A. Ahmed, K. Yasumoto, N. Shibata & T Kitani, HDAR: Highly distributed adaptive service replication for manets, *IEICE Transactions on Information and Systems* **94** (2011), 91–103.
[20] M. Satyanarayanan, Pervasive computing: Vision and challenges, *IEEE Personal Communications* **8** (2001), 10–17.
[21] L. Kung-Kiu & Z. Wang, Software component models, *IEEE Transactions on Software Engineering* **33** (2001), 709–724.
[22] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, L. Viennot, Optimized link state routing protocol (OLSR), *Network Working Group* (2003).
[23] T. Ruiz-López, C. Rodríguez-Domínguez, S. Ochoa, J.L. Garrido, Mdubi: A model-driven approach to the development of self-adaptive services for ubiquitous systems, *Sensors* (2014), in press.
[24] P. Krill, Make way for SOA 2.0, *http://www.infoworld.com/t/architecture/make-way-soa-20-420* (2006).
[25] C. Rodríguez-Domínguez, K. Benghazi, M. Noguera, J.L. Garrido, M.L. Rodríguez, T. Ruiz-López, A communication model to integrate the request-response and the publish-subscribe paradigms into ubiquitous systems, *Sensors* **12** (2012), 7648–7668.
[26] R. Guerraoui & A. Schiper, Software-based replication for fault tolerance, *Computer* **30** (1997), 68–74.
[27] A. Fuggetta, G.P. Picco, G. Vigna, Understanding code mobility, *IEEE Transactions on Software Engineering* **24** (1998), 342–361.